Web Development: It’s Not a Bug, It’s a Feature

John McCardle

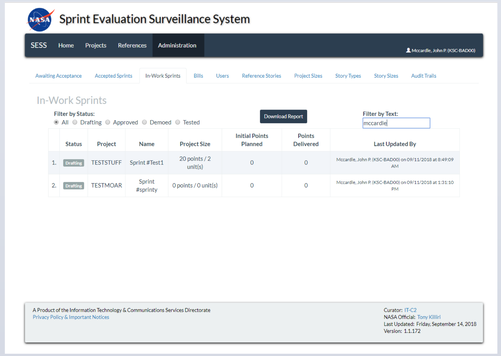
IT-C2 – Application Engineering and Operations Branch

SESS / CFC Applications

For a new developer in any organization, starting from scratch can be daunting. Instead, I’ve been assigned to fix bugs or build enhancements for some existing software. This has given me the chance to learn my branch’s coding standards and tools of choice without the pressure of creating a brand new application.

CFC Administration Application

The Combined Federal Campaign (CFC) runs every year to collect charitable donations from civil servants. The CFC application tracks “Quality Contacts” to help Key Workers locate uncontacted employees. The application also presents tables of data and an appealing thermometer graph to keep users informed.



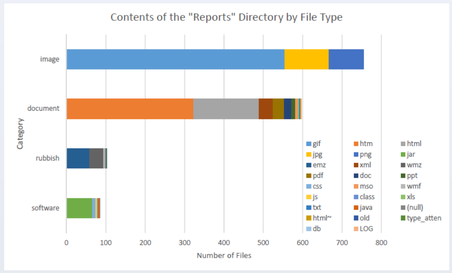
Sprint Evaluation Surveillance System

The KIAC contract is responsible for much of the software development on center, and it is the role of my branch to assign the work appropriately and then monitor its completion.

The Sprint Evaluation Surveillance System (SESS) is a web application to track all software development sprints currently in progress. During and after each sprint, development goals can be tracked, changed, and reviewed. This was an excellent choice of software assignment for a new member, because not only am I developing a piece of software, but also learning the processes our branch is responsible for.

Website Migration To Sitecore

Nothing is permanent. Though digital products may have an immaterial quality to them, software and websites have a shelf life. Websites only exist to support missions, and when a mission outlasts its web technologies, a migration is necessary. During a migration, we assess the website and move it to a new platform. I have been assessing a website for migration to Sitecore, KSC’s content management system.



Reports Detected by Web Scraper (Above)

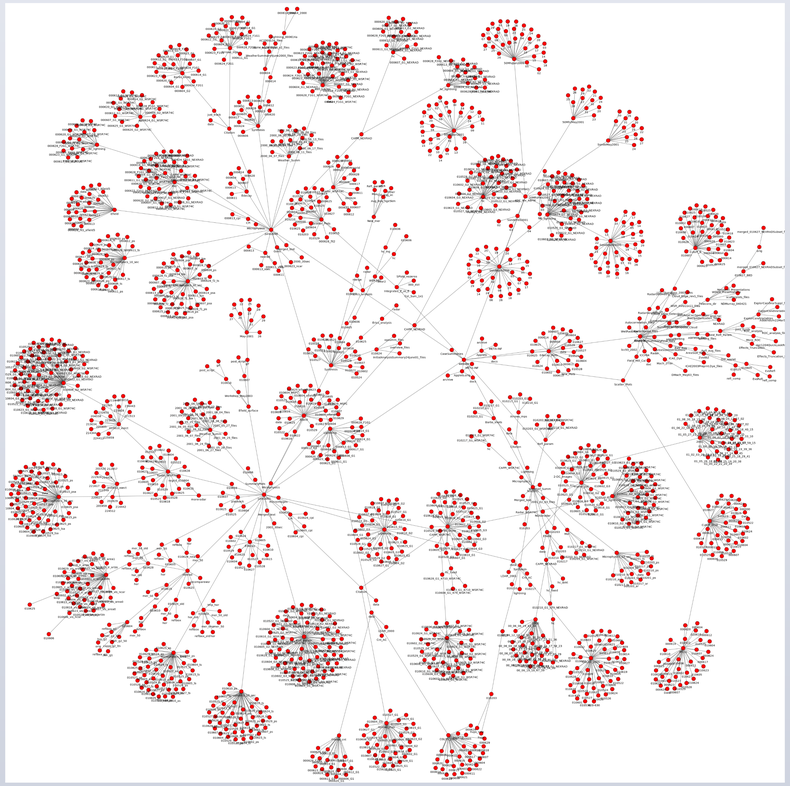
My web scraper program detected over 1,500 files that went undetected by earlier site analysis.

Web Scraping is the technique of using a computer program other than a web browser to access websites, in order to collect all or some of the data. In this case, I used a program called a web scraper or *spider* to collect a complete listing of the files and directories of one KSC website. This information helps our customer to make determinations about what files are still in use, which files need to be archived, and which can be safely discarded.

```python
def analyse_link(href):
"""Determines what file type or link a URL points to."""
        if href.endswith('/'):
                ltype = "dir"
        elif href.startswith('mailto:'):
                ltype = "email"
        else:
                ltype = "file"
        return ltype

def links_on_page(url):
"""Returns a list of bare URLs on the given page. Full URL required."""
        r = requests.get(url)
        soup = BeautifulSoup(r.text, 'html.parser')
        #print(f"page {url} {len(r.text)} bytes")
        return [l['href'] for l in soup.find_all('a') if
        analyse_link(l['href']) == 'dir']

def isolated_page(url):
"""Splits a URL into components and returns the final portion and the
directory depth. Either full or relative URLs will work, but don't compare
the depth of full and partial"""
        isolate = [s for s in url.split('/') if not len(s) == 0]
        if len(isolate) == 0: return ('/', 0)
        else: return (isolate[-1], len(isolate))

def analyse_page(start_url, base_page):
"""Recursively follows relative URLs on a page to map out all
subdirectories."""
        url = base_page + start_url
        this_page, depth = isolated_page(start_url)
        directory = {}
        links = links_on_page(url)
        #print(f"page: {url} -> '{this_page}' links: {len(links)}")
        for link in links:
                l, d = isolated_page(link)
                if d < depth:
                        #print(f"Skipping link to {l}")
                        continue
                #print(f"Found sublink[{d}] to {l}")
                directory[l] = analyse_page(link)
        return directory

import json
map_json = json.dumps(page_map, indent=4) #indent for pretty printing
with open("page_map.json", "w") as f:
        f.write(map_json)
```

Excerpt of Python Code (Above)

This web scraper, written in Python, does not take much code. The secret to its effectiveness is running the code recursively over the target website’s 1,800+ subdirectories.



Website Directory Tree (Left)

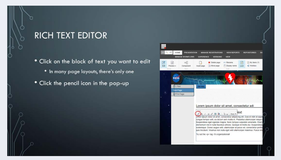
This tree of a KSC website was generated with data gathered by the web scraper. Every vertex represents a directory, and the connecting lines indicate a directory’s contents. The web scraper recursively explores every subdirectory until the entire site has been indexed. This particular website had 1,699 subdirectories.

Following my investigation, we were able to categorize the contents of the website to separate relevant content from historical and out of date content. For a site of this magnitude, it’s crucial that we only spend effort migrating content that is still of value to the organization.

Sitecore Training

Teaching others to use our systems is a crucial element of web development. I’ve already begun sharing my new expertise with Sitecore.

I wrote a Sitecore training slideshow and user guide. After heavy editing on the part of my mentor, our new documents were approved and I led the first Sitecore training on center in several months.



Sitecore Training Slide (Left)

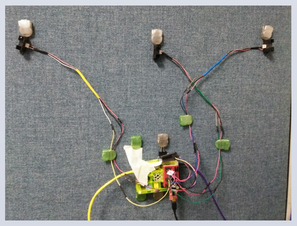
Every step a new Sitecore editor needs to follow to change their organization’s website is shown in the course.

Sitecore Quick Help Guide (Right)

The Quick Help Guide explains every element of the training in prose form, which should help new Sitecore editors remember their training or brush up between infrequent editing.

Fun

The office can be a quiet environment: so quiet that any minor disturbance becomes an alarming experience. To prevent my co-workers from startling me when entering my cubicle, I built this entry detector. Different sound effects are played for entry and exit.



Three infrared distance sensors scan for objects passing between them and the opposite wall. When the beam is broken, software tracks the sequence of interruptions to determine the direction of travel.

Acknowledgements

I’d like to thank my mentor, Laurie Brown, for guiding me through my first weeks at NASA and showing me the ropes of Sitecore. Additionally, everyone in IT-C2 has had unlimited patience with me, and great generosity with their knowledge.